SOFT MAIL

สร้างภาพกราฟิก ด้วยเมาส์

อยากวาดรูปบนคอมพิวเตอร์ ไม่ใช่เรื่องยาก เพียงแต่มีเจ้าหนูหางยาวตัวนี้ คุณก็สามารถสร้างจินตนาการบนภาพกราฟิกได้อย่างสบาย

อุปกรณ์ประกอบการใช้งานในเครื่องคอมพิวเตอร์อีกอย่างหนึ่ง คือ เมาส์ (mouse) หรือเจ้าหนูหางยาว ซึ่งในปัจจุบันจะพบว่าเข้ามามีบทบาทในโปรแกรมสำเร็จหลาย ๆ โปรแกรม เช่น ไมโครซอฟต์วินโดวส์ หรือโปรแกรมในการสแกน (scan) ภาพต่าง ๆ เมาส์ที่ผู้เขียนใช้นี้มีปุ่มเลือกอยู่ 3 ปุ่มด้วยกัน เมื่อพลิกดูด้านล่างพบว่ามีสวิตช์เล็ก ๆ อยู่ตัวหนึ่งโดยเขียนข้อความว่า P<-->M ซึ่งตามคู่มือระบุว่าเป็นการจัดตั้งเมาส์ให้ตรงกับความต้องการคือ เมื่อเลือก P จะหมายถึงจัดตั้งเป็น PC mouse และเมื่อเลือก M จะเป็น Microsoft mouse เราจะเสียบเมาส์นี้เข้ากับส่วนต่อเชื่อมกับเครื่องคอมพิวเตอร์ที่เรียกกันว่า COM1 หรือ COM2 (ซึ่งเป็นพอร์ตอนุกรม) ก่อนจะเข้าสู่โปรแกรมใด ๆ ที่ใช้เมาส์ เราจะต้องใช้ไฟล์จัดการของเมาส์ที่ให้มาพร้อมกับเมาส์ที่เราซื้อ ซึ่งอาจจะเป็นไฟล์ชื่อ mouse.com (ดูชื่อไฟล์จากคู่มือเมาส์ของท่านอีกที) โดยป้อนคำสั่งผ่านดอสดังนี้

```
A>mouse -p1 เมื่อต่อกับ COM1 และตั้งสวิตช์เป็น PC mouse
A>mouse -p2 เมื่อต่อกับ COM2 และตั้งสวิตช์เป็น PC mouse
A>mouse -m1 เมื่อต่อกับ COM1 และตั้งสวิตช์เป็น Microsoft mouse
A>mouse -m2 เมื่อต่อกับ COM2 และตั้งสวิตช์เป็น Microsoft mouse
```

พารามิเตอร์ (-p1, -p2, -m1, -m2) นี้จะต้องดูค่าที่ถูกต้องจากคู่มืออีกที เพราะบางไฟล์อาจจะจัดตั้งให้ที่ COM1 ในทันทีที่เรียกใช้ไฟล์ เช่น อีกไฟล์หนึ่งที่ผู้เขียนมีชื่อ gmouse.com จะป้อนพารามิเตอร์อีกแบบหนึ่ง

ตรวจการจัดตั้งเมาส์

เมื่อได้จัดตั้งเมาส์ด้วยไฟล์ mouse.com เรียบร้อยแล้วนั้น เราสามารถตรวจสอบว่ามีการติดตั้งระบบของเมาส์หรือยัง โดยการตรวจผ่านอินเตอร์รัพต์หมายเลข 33 (ฐานสิบหก) หรือ 51 (ฐานสิบ) ที่ใช้ควบคุมเมาส์ ซึ่งจะสามารถตรวจการติดตั้งเมาส์, แสดงเคอร์เซอร์ของเมาส์, หยุดแสดงเคอร์เซอร์ของเมาส์, กำหนดขอบเขตของเมาส์, กำหนดจุดเริ่มต้นของเมาส์, ตรวจหาจุดที่เคอร์เซอร์อยู่ ฯลฯ ได้

สำหรับบทความนี้ผู้เขียนขอเสนอแนวในการใช้เมาส์เพื่อวาดภาพกราฟิก โดยผู้เขียนทดสอบโปรแกรมนี้ผ่านจอสีและจอวีจีเอโมโนโครม (จัดตั้งให้เป็นกราฟิกชนิดวีจีเอ-ไม่ใช่วีจีเอ-โมโนโครม) ขอยกตัวอย่าง การใช้กราฟิกชนิดวีจีเอในการ์ดภาษาไทยของ IRC ซึ่งจะมีไฟล์จัดการชื่อ svm89.com สามารถจัดตั้งได้ดังนี้

c>svm89 vga เพื่อจัดให้ได้กราฟิกแบบ vga (ผู้ใช้ยนใช้ในโหมดนี้ในกรณีที่เป็นจอวิจัอโมโนโครม)

c>svm89 ega เพื่อจัดให้ได้กราฟิกแบบ ega

c>svm89 cga เพื่อจัดให้ได้กราฟิกแบบ cga

c>svm89 herc เพื่อจัดให้ได้กราฟิกแบบโมโนโครม (ซึ่งจะรันไฟล์ที่จอโมโนโครมธรรมดาได้ทันที)

สำหรับผู้ที่ใช้จอสีอยู่แล้วก็ไม่ต้องเรียกใช้ไฟล์นี้เพราะทำงานบนจอสีอยู่แล้ว โปรแกรมในรูปที่ 1 จะเป็นชุดคำสั่งเข้ากราฟิกและการตรวจการติดตั้งเมาส์ มีรายละเอียดดังนี้

1. กำหนดตัวแปร mouse, mono, vgamono เพื่อใช้ตรวจสอบค่าต่าง ๆ ตัวแปร Regs มีไว้รองรับค่าที่อ่านได้ผ่านอินเตอร์รัพต์ 33H ส่วน xasp,yasp ใช้คำนวณหารัศมีของวงกลมให้ตรงกับความจริง

2. ชุดคำสั่ง set_color เพื่อกำหนดสีของการวาดภาพให้อยู่ในขอบเขตของจอภาพที่ใช้ (ถ้าเป็นจอสีไม่ต้องมีชุดคำสั่งนี้ก็ได้ ใช้ setcolor ได้เลย)

3. ชุดคำสั่ง opengraph เพื่อเข้าสู่กราฟิกโหมด ในที่นี้ผู้เขียนกำหนดให้เป็นกราฟิกขนาดที่มีค่าตามแกน y ไม่เกิน 350 แถว

4. ชุดคำสั่ง CheckMouse ตรวจการติดตั้งเมาส์ ซึ่งหมายถึงการได้รันไฟล์ที่เป็นไดรเวอร์ของเมาส์แล้วหรือยัง โดยกำหนดฟังก์ชัน 0 คือ regs.AX=0 แล้วเรียกใช้อินเตอร์รัพต์หมายเลข 33H ซึ่งจะทำให้ได้ค่าตัวแปร regs คืนกลับมาใช้ตรวจสอบส่วนที่เกี่ยวข้องกับเมาส์ได้ดังนี้

regs.AX ตรวจว่าติดตั้งเมาส์หรือยัง ถ้าค่าเป็น 0 จะยังไม่ได้ติดตั้งเมาส์

regs.BX ให้จำนวนปุ่มของเมาส์ สำหรับเมาส์ที่ผู้เขียนมีอยู่นั้นมี 3 ปุ่ม เมื่อจัดตั้งให้เป็น PC mouse จะรายงานว่ามี 3 ปุ่ม แต่เมื่อจัดตั้งให้เป็น Microsoft mouse จะรายงานว่ามีเพียง 2 ปุ่มเท่านั้น (ปุ่มกลางไม่ทำงาน) ในตัวอย่างโปรแกรมทั้งหมดนี้ผู้เขียนจัดตั้งเป็น PC mouse จึงทำให้ใช้งานปุ่มได้ทั้ง 3 ปุ่ม ชุดคำสั่งจะรายงานต่าง ๆ ตามที่กำหนดไว้ และถ้า regs.AX<>0 ก็ให้ตัวแปร mouse เป็นจริง

```
 1: PROGRAM Mouse_Editor;
 2: uses
 3:   Dos, Crt,graph;

 4: var Mouse,Mono,VgaMono:boolean;
 5:     color:byte;    regs:Registers;
 6:     Pic:pointer;
 7:     Ch:char;
 8:     Xasp,yasp:word;

 9: PROCEDURE Set_color(Colors:byte);
10: begin
11:    if Mono then Colors:=1;
12:    if VgaMono then Colors:=3;
13:    setcolor(Colors);
14:    Color:=Colors;
15: end;

16: PROCEDURE OpenGraph;
17: var grdriver,grmode,Graphck,n,k:integer;
18: begin
19:    Mono:=false; VgaMono:=false;
20:    detectgraph(grdriver,grmode);
21:    case grdriver of
22:      5:VgaMono:=true;
23:      7:Mono:=true;
24:      9:grmode:=1;
25:    end;
26:    initgraph(grdriver,grmode,'');
27: end;

28: PROCEDURE CheckMouse;
29: var st:string;
30: begin
31:    regs.AX:=0; Intr($33,regs);  set_color(14);
32:    if regs.AX=0 then outtextxy(100,10,'Can not find mouse driver')
33:       else
34:         begin
35:           outtextxy(100,10,'mouse driver Installed');
36:           str(regs.BX,st);  outtextxy(280,10,'('+st+' buttons)');
37:           Mouse:=true;
38:         end;
39: end;
```

รูปที่ 1 ชุดคำสั่งเข้าสู่กราฟิกโหมดและการตรวจการจัดตั้งระบบของเมาส์

การตรวจค่าอื่น ๆ ของเมาส์

เมื่อเราพบว่าได้มีการติดตั้งเมาส์เข้ากับระบบแล้ว เราก็สามารถสร้างชุดคำสั่งตรวจสอบในเรื่องอื่น ๆ โดยเป็นการอาศัยอินเตอร์รัพต์ 33H โดยกำหนดฟังก์ชันในการตรวจสอบต่าง ๆ กันออกไปดังนี้

ฟังก์ชัน 1 (regs.AX=1) เพื่อแสดงเคอร์เซอร์ของเมาส์

ฟังก์ชัน 2 (regs.AX=2) เพื่อหยุดแสดงเคอร์เซอร์ของเมาส์

ฟังก์ชัน 3 (regs.AX=3) เพื่อตรวจการกดปุ่มของเมาส์

ฟังก์ชัน 4 (regs.AX=4) เพื่อหาตำแหน่งของเมาส์ (หาค่าในแกน x และแกน y)

ฟังก์ชัน 7 (regs.AX=7) เพื่อกำหนดขอบเขตที่ให้เมาส์เคลื่อนที่ได้ในแนวนอน (แกน x)

ฟังก์ชัน 8 (regs.AX=8) เพื่อกำหนดขอบเขตที่ให้เมาส์เคลื่อนที่ได้ในแนวตั้ง (แกน y)

โปรแกรมในรูปที่ 2 จะเป็นชุดคำสั่งทั้งหมดในการจัดการเกี่ยวกับค่าต่าง ๆ ของเมาส์ มีรายละเอียดดังนี้

1. บรรทัดที่ 40-43 ชุดคำสั่งแสดงเคอร์เซอร์ของเมาส์ ซึ่งจะให้สีเป็นสีขาว หากประสงค์ให้แสดงสีของเคอร์เซอร์ของเมาส์เป็นสีอื่นก็ให้ใช้คำสั่ง SetPalette ช่วย เช่น SetPalette(15,13); จะได้สีเคอร์เซอร์เป็นสีม่วงหรือ SetPalette(15,14); จะได้เคอร์เซอร์สีเหลืองแทน (จะมีผลต่อสีของส่วนอื่น ๆ ในเมนูด้วย) ลักษณะเคอร์เซอร์ของเมาส์ดูได้จากรูปที่ 5 ซึ่งจะอยู่ต่อท้ายคำว่าสวัสดี

2. บรรทัดที่ 44-47 ชุดคำสั่งหยุดการแสดงเคอร์เซอร์โดยอาศัยฟังก์ชัน 2

3. บรรทัดที่ 48-52 ชุดคำสั่งกำหนดตำแหน่งเริ่มต้นของเคอร์เซอร์ใช้ฟังก์ชัน 4 โดยการกำหนดค่าให้ตัวแปร regs.CX และ regs.DX แล้วเรียกอินเตอร์รัพต์ 33H ซึ่งจะทำให้เคอร์เซอร์ของเมาส์ไปปรากฏ ณ จุดที่กำหนดผ่านตัวแปร x,y

4. บรรทัดที่ 53-57 ชุดคำสั่งกำหนดขอบเขตของเคอร์เซอร์ในแนวนอน ซึ่งหมายถึงการกำหนดให้เคอร์เซอร์เคลื่อนย้ายในแนวนอนได้ซ้ายสุดและขวาสุด โดยกำหนดที่อาร์กิวเมนต์ L,R และฟังก์ชัน 7 โดยผู้เขียนโปรแกรมต้องกำหนดเองตามความต้องการ

```
40: PROCEDURE MouseOn;                  { กำหนดให้แสดงเคอร์เซอร์ของ mouse }
41: begin
42:    regs.AX:=1; Intr($33,regs);
43: end;

44: PROCEDURE MouseOff;                 { กำหนดไม่ให้แสดงเคอร์เซอร์ของ mouse }
45: begin
46:    regs.AX:=2; Intr($33,regs);
47: end;

48: PROCEDURE SetPosition(X,Y:word);    { กำหนดตำแหน่งเริ่มต้น }
49: begin
50:    regs.CX:=X; regs.DX:=Y;
51:    regs.AX:=4; Intr($33,regs);
52: end;

53: PROCEDURE SetLeftRight(L,R:word);   { ตั้งขอบเขตด้านซ้ายและขวา }
54: begin
55:    regs.CX:=L; regs.DX:=R;
56:    regs.AX:=7; Intr($33,regs);
57: end;

58: PROCEDURE SetUpDown(U,D:word);      { ตั้งขอบเขตบนและล่าง }
59: begin
60:    regs.CX:=U; regs.DX:=D;
61:    regs.AX:=8; Intr($33,regs);
62: end;

63: PROCEDURE FindXY(Var x,y:integer);  { หาตำแหน่งในแกน x,y }
64: begin
65:    regs.AX:=4; Intr($33,regs);
66:    x:=regs.CX;  y:=regs.DX;
67: end;
```

รูปที่ 2 ชุดคำสั่งในการจัดการเกี่ยวกับเมาส์ทั้งหมด

5. บรรทัดที่ 58-62 ชุดคำสั่งกำหนดขอบเขตของเคอร์เซอร์ในแนวตั้ง

ให้สังเกตชุดคำสั่งตามข้อ 3-5 นี้จะต้องกำหนดค่า regs.CX, regs.DX ก่อนที่จะเรียกอินเตอร์รัพต์ ซึ่งหมายถึงการนำค่าที่กำหนดนี้ส่งไปเก็บไว้ที่อินเตอร์รัพต์นั้น ๆ

6. บรรทัดที่ 63-67 ชุดคำสั่งหาค่าของ x,y เพื่อนำมาใช้ในงานกราฟิกเพื่อวาดภาพตามความต้องการของผู้พัฒนาโปรแกรม ซึ่งจะเป็นการนำค่า regs.CX และ regs.DX มาส่งให้กับตัวแปร x,y ที่เรียกเข้ามา ขอให้สังเกตว่าผู้เขียนกำหนดพารามิเตอร์ x,y เป็น var ซึ่งหมายถึงว่าจะส่งค่าที่หาได้คืนให้กับตัวแปรที่เรียกเข้ามา

อธิบายให้เข้าใจได้ง่าย ๆ ดังนี้

ถ้าใช้คำสั่ง findXY(a,b); จะทำให้ได้ตัวแปร a=x=regs.CX และตัวแปร b=y=regs.DX หรือใช้คำสั่ง findXY(cx,cy); จะได้ตัวแปร cx=x=regs.CX และตัวแปร cy=y=regs.DX เป็นต้น

วาดภาพด้วยเมาส์

ผู้อ่านอาจจะสงสัยในเรื่องการตรวจว่าปุ่มใดของเมาส์ถูกกดนั้นทำอย่างไร คงต้องอดใจรอสักนิดหนึ่ง เพราะช่วงต่อไปนี้ผู้เขียนจะเปิดเผยถึงการตรวจสอบการกดปุ่มของเมาส์ เพื่อให้โปรแกรมนี้ทันสมัยตามยุคของจอสี ผู้เขียนจึงขอเสนอแนวทางการสร้างภาพกราฟิกด้วยเมาส์เพื่อเป็นตัวอย่างหนึ่งชุดคำสั่ง ซึ่งจะกำหนดการวาดภาพไว้เป็นตัวอย่าง 5 แบบ คือ

1. การลงจุดขนาด 1 จุด (Mode =1)

2. การลงจุดขนาด 3×3 จุด (Mode=2)

3. การวาดรูปวงกลม (Mode=3)

4. การวาดรูปสี่เหลี่ยมผืนผ้า (Mode=4)

5. การลากเส้นตรง (Mode=5)

โดยในเมนูของการวาดภาพผู้เขียนได้กำหนดตารางแสดงส่วนของรูปแบบพื้นจำนวน 11 แบบ (เพื่อวาดรูปสี่เหลี่ยมด้วยพื้นหลาย ๆ แบบ) ไว้ทางด้านซ้ายของจอภาพ กำหนดตารางแสดงสีจำนวน 15 สีไว้ทางด้านขวาของจอภาพ และแสดงภาพของการเลือกรูปแบบการวาดภาพ 5 แบบ (ตามที่กล่าวข้างต้น) ไว้ทางด้านขวาต่ำลงมาจากส่วนของรูปแบบพื้น โดยมีกรอบสี่เหลี่ยมขนาดเล็กกว่าขนาดของจอภาพให้ใช้ในการวาดภาพ กรอบนี้มีไว้ให้ผู้อ่านมองเห็นการทำงานในการกำหนดขอบเขตซ้ายขวาและบนล่างของเคอร์เซอร์เมาส์

ผู้เขียนจงใจให้ขนาดของตารางในการเลือกรูปแบบของการวาดภาพ 5 แบบนั้นอยู่ต่ำกว่าด้านล่างของกรอบในการวาดภาพ (ขอให้ดูรูปที่ 5 ประกอบ) ทั้งนี้เพื่อต้องการแสดงให้เห็น

รูปที่ 3 ชุดคำสั่งการวาดภาพโดยใช้เมาส์

```
68: PROCEDURE PicByMouse;
69: Const Ypatt:array[1..16] of word=
70:         (30,48,66,84,102,120,138,156,174,192,210,228,246,264,282,300);
71:       ModeST:array[1..5] of string[13]=
72:          ('Plot 1 pixel','Bar 3x3 pixel','Cirele','Bar','Line');
73: var   x,y,Cx,Cy,CxOld,CYold,Radius,Ypic:integer;
74:       cc,No,patt,Mode,CMode,xx:byte;
75:       x1,y1:real;

76: begin
77:     outtextxy(120,320,'Mouse Editor By Arthan Sattayarak Lampang');
78:     color:=14;  Patt:=1;        Mode:=1;
79:     set_color(color);           rectangle(40,30,580,300);
80:     SetLeftRight(10,608);       SetUpDown(31,299);
81:     SetPosition(270,165);       CxOld:=270; CyOld:=165;
82:     set_color(12); cc:=getcolor;
83:     for x:=1 to 15 do
84:       begin
85:         setfillstyle(x,cc);
86:         if x<12 then bar(10,Ypatt[x],30,Ypatt[x]+15);
87:         setfillstyle(1,x);
88:         bar(590,Ypatt[x],620,Ypatt[x]+15);
89:         set_color(15);
90:         rectangle(590,Ypatt[x],620,Ypatt[x]+15);
91:         set_color(10); if x>11 then set_color(14);
92:         rectangle(10,Ypatt[x],30,Ypatt[x]+15);
93:       end;
94:     rectangle(10,Ypatt[16],30,Ypatt[16]+15);
95:     set_color(12);       rectangle(5,Ypatt[12]-1,35,Ypatt[16]+17);
96:     set_color(10);
97:     outtextxy(0,20,'pattern'); outtextxy(585,20,'color');
98:     outtextxy(5,325,'mode');   outtextxy(385,20,ModeST[Mode]);
99:     putPixel(20,Ypatt[12]+7,getcolor);
100:    bar(19,Ypatt[13]+6,22,Ypatt[13]+9);
101:    circle(20,Ypatt[14]+7,7);
102:    bar(15,Ypatt[15]+3,26,Ypatt[15]+12);
103:    line(15,Ypatt[16]+8,26,Ypatt[16]+8);
104:    set_color(14);          setfillstyle(1,getcolor);
105:    Ypic:=Ypatt[12]+2;      getimage(12,Ypic,28,Ypic+11,pic^);
106:    putimage(12,Ypic,pic^,4);   MouseOn;
107:  repeat
108:      if Regs.CX<29 then SetUpDown(31,313)
109:                    else  SetUpDown(31,299);
110:      regs.AX:=3; Intr($33,regs);    { ตรวจการกดปุ่ม }
111:     case regs.BX of
112:      1:{ เมื่อกดปุ่มด้านซ้าย จะลงจุด}
113:        begin
114:          FindXY(x,y);
115:          CMode:=0;
116:          if x>40 then Cmode:=1;
117:          if x>=590 then Cmode:=2;
118:         Case Cmode of
119:          1:begin
120:              if x<580 then
121:                begin
122:                  case Mode of
123:                    1:putpixel(x,y,color);
124:                    2:begin   MouseOff;
125:                      setfillstyle(Patt,color);
126:                      bar(x,y,x+2,y+2);
127:                            MouseOn;
128:                    end;
```

รูปที่ 3 (ต่อ)

```
129:                        3:begin
130:                            MouseOff;
131:                             x1:=abs(x-cx); y1:=yasp/xasp*abs(y-cy);
132:                             Radius:=round(sqrt(sqr(x1)+sqr(y1)));
133:                             setviewport(41,31,579,299,clipon);
134:                             circle(Cx-41,Cy-31,Radius);
135:                             setviewport(0,0,getMaxx,GetMaxy,clipon);
136:                             MouseOn;
137:                           end;
138:                        4:begin
139:                            MouseOff;
140:                            Bar(Cx,Cy,x,y);
141:                            rectangle(Cx,Cy,x,y);
142:                            MouseOn;
143:                          end;
144:                        5:begin
145:                            MouseOff;
146:                            line(Cx,Cy,x,y);
147:                            MouseOn;
148:                          end;
149:                      end;  { Case Mode }
150:                  end { x<580 }
151:              end;   { Case 1 }
152:            0,2:begin
153:                xx:=1;  No:=0;
154:                repeat
155:                 if (y>=Ypatt[xx]) and (y<=Ypatt[xx]+15) then No:=xx;
156:                 inc(xx);
157:                until (xx>16) or (No<>0);
158:                if no<>0 then
159:                  begin
160:                     Case CMode of
161:                      0:if No<12 then
162:                           begin setfillstyle(No,color);
163:                                 patt:=no;
164:                           end
165:                          else
166:                           begin  MouseOff;
167:                            putimage(12,Ypic,pic^,0);
168:                            Mode:=No-11;
169:                            Ypic:=Ypatt[No]+2;
170:                            getimage(12,Ypic,28,Ypic+11,pic^);
171:                            putimage(12,Ypic,pic^,4);
172:                            setfillstyle(1,0);
173:                            bar(380,0,500,27);
174:                            setfillstyle(patt,color);
175:                            outtextxy(385,20,ModeST[Mode]);
176:                            Cx:=CxOld; Cy:=CyOld;
177:                                  MouseOn;
178:                           end;
179:                      2:begin
180:                          if No<16 then
181:                          begin
182:                           Color:=No;  set_color(color);
183:                          end;
184:                        end;
185:                    end;
186:                end;   { if no<>0}
187:              end;   { 0,2 }
188:          end;   { Case CMode }
189:        end;  { regs.BX=1 }
190:      4:begin
```

ว่าการกำหนดขอบเขตในการเลื่อนเคอร์เซอร์ให้อยู่ในส่วนที่เราต้องการนั้นทำได้อย่างไร เมื่อบางส่วนมีขนาดขอบเขตไม่เท่ากัน ส่วนนี้ผู้เขียนอยากให้ผู้อ่านลองพิจารณาศึกษาจากโปรแกรมให้ออกก่อนที่จะอ่านคำอธิบาย

หลักในการวาดภาพมีแนวคิดดังนี้

1. ผู้ใช้โปรแกรมวาดภาพต้องกดปุ่มซ้ายของเมาส์เมื่อต้องการวาดภาพ, เลือกรูปแบบพื้น หรือเลือกสีโดยโปรแกรมจะตรวจให้เองว่ากำลังอยู่ในส่วนใดของจอภาพเพื่อที่จะทำงานได้อย่างถูกต้อง

2. ในกรณีที่จะวาดรูปวงกลม, รูปสี่เหลี่ยม หรือลากเส้น สามารถกำหนดจุดศูนย์กลาง, มุมของรูปสี่เหลี่ยม (มุมแรก) หรือจุดเริ่มต้นของเส้นตรงด้วยการกดปุ่มกลางของเมาส์

3. การวาดภาพตามข้อ 2 นั้น หากไม่มีการเปลี่ยนจุดตามที่กล่าวในข้อ 2 ก็จะสามารถใช้ปุ่มซ้ายสร้างภาพได้เรื่อย ๆ ไป

4. ในระหว่างการวาดภาพนั้นสามารถเลื่อนเมาส์ไปเลือกรูปแบบของพื้นและสี (โดยเลือกสีก่อนแล้วไปเลือกรูปแบบพื้น) และเลือกสีเพียงอย่างเดียวได้ตลอดเวลาโดยการกดปุ่มซ้ายมือเพียงอย่างเดียว

5. สามารถเลื่อนเมาส์ไปเลือกรูปแบบการวาดภาพ 5 แบบที่อยู่ทางมุมล่างซ้ายได้ โดยกดปุ่มซ้ายเมื่อเคอร์เซอร์ (ปลายลูกศร) ปรากฏในกรอบของส่วนที่ต้องการเลือก

6. เมื่อต้องการยุติการวาดภาพก็ให้กดปุ่มขวาของเมาส์ ก็จะจบการทำงานทั้งหมด

นำแนวคิดนี้มาเขียนชุดคำสั่งได้ดังรูปที่ 3 ซึ่งมีความยาวถึง 136 บรรทัด มีรายละเอียดดังนี้

รูปที่ 3 (ต่อ)

```
191:              if Mode in [3..5] then
192:                begin
193:                 FindXY(x,y);
194:                 if (x>40) and (x<580) then
195:                    begin
196:                      Cx:=x; Cy:=y;  CxOld:=Cx;  CyOld:=Cy;
197:                      putpixel(x,y,color);
198:                    end;
199:                end;
200:              end;
201:        end;   { Case regs.BX }
202:     until regs.BX=2;      { จนกว่าจะกดปุ่มด้านขวา }
203: end;                                                    ■
```

1. บรรทัดที่ 68-75 กำหนดค่าคงที่ไว้ตรวจสอบตำแหน่งของการเลือกรูปแบบพื้น, โหมดการวาดภาพและสีของจุดหรือเส้นที่จะวาด (Ypatt) โดยจะนำไปคำนวณหาตำแหน่งเมื่อมีการกดปุ่มซ้ายของเมาส์ ค่าคงที่ ModeST เป็นข้อความที่จะนำไปแสดงให้ทราบว่ากำลังอยู่ในโหมดใดของการวาดภาพ ส่วนตัวแปรอื่น ๆ นั้นกำหนดไว้เพื่อนำไปใช้งานในชุดคำสั่ง

2. บรรทัดที่ 76-82 แสดงข้อความให้ทราบว่าเป็นเรื่องใด กำหนดสีเริ่มต้น วาดกรอบสี่เหลี่ยมไว้ใช้เป็นขอบเขตในการวาดรูป (ส่วนนี้จะถือว่าเป็น Cmode=1) กำหนดขอบเขตซ้ายขวาและบนล่างของตัวเคอร์เซอร์ (บรรทัดที่ 80) ซึ่งจะเห็นว่าค่าในแกนตั้งจะน้อยกว่ากรอบที่กำหนดอยู่ 1 จุด ส่วนค่าในแกนนอนจะกำหนดอยู่ที่จุด 10 และ 608 ซึ่งเป็นด้านซ้ายสุดของตารางรูปแบบพื้นและด้านขวาของตารางสีตามลำดับ

บรรทัดที่ 81 จะกำหนดตำแหน่งเริ่มต้นให้เคอร์เซอร์และกำหนดค่าเริ่มต้นของจุดศูนย์กลางของวงกลมหรือมุมและของสี่เหลี่ยมหรือจุดแรกของเส้นตรง

3. บรรทัดที่ 83-106 ส่วนของการแสดงตารางเพื่อกำหนดรูปแบบพื้น, โหมดการวาดภาพ และตารางสี โดยมีข้อความกำกับให้ทราบว่าตารางใดเป็นส่วนใดของเมนูที่จะเลือกการทำงาน

บรรทัดที่ 99-105 จะแสดงเมนูของการวาดภาพแบบ ลงจุด วงกลม สี่เหลี่ยม และเส้นตรงแบบง่าย ๆ ผู้อ่านอาจเพิ่มเติมคำสั่งให้แสดงภาพที่สวยกว่านี้ก็ได้ โดยจะแสดงส่วนของการลงจุดแบบกลับขาวดำเพื่อให้ผู้ใช้รู้ว่าจะเริ่มต้นที่การลงจุด (มีข้อความว่า putpixel ปรากฏที่ด้านบนขวาของกรอบการวาดภาพ) และกำหนดให้แสดงเคอร์เซอร์ของเมาส์ในบรรทัดที่ 106 เพื่อเป็นการเริ่มต้นการวาดรูป

4. บรรทัดที่ 108-110 ตรวจการกดปุ่มของเมาส์โดยอาศัยฟังก์ชัน 3 ของอินเตอร์รัพต์ 33H เมื่อทำการตรวจ regs.BX ก็จะทราบว่ามีการกดปุ่มใดของเมาส์ (เมื่อจัดตั้งเป็น PC mouse) ดังนี้

```
regs.BX = 1  กดปุ่มด้านซ้าย
regs.BX = 2  กดปุ่มด้านขวา
regs.BX = 4  กดปุ่มกลาง
```

โดยการตรวจที่ regs.CX ก็จะทำให้ทราบได้ว่าเมาส์ได้ถูกเลื่อนไปอยู่ที่ค่าใดในแกนนอน (แกน x) ถ้าพบว่าค่า regs.CX<29 (อยู่ในแนวของตารางส่วนของการเลือกรูปแบบและโหมดการวาดภาพส่วนนี้คือ ส่วนที่มีค่า Cmode=0) ก็จะกำหนดให้ค่าในแนวดิ่งด้านล่างสุดเป็น 313 เพื่อที่จะเลื่อนลงมาเลือกการวาดภาพแบบเส้นตรงได้ แต่ถ้าเลื่อนเมาส์ไปทางขวาพ้นขอบเขตของตารางซ้ายมือนี้ก็จะเลื่อนด้านล่างมาที่ 299 ตามเดิม

5. เมื่อมีการกดปุ่มซ้ายมือของเมาส์จะมีการทำงานตามบรรทัดที่ 112-189 ดังรายละเอียดต่อไปนี้

6. บรรทัดที่ 114-117 จะทำการตรวจหาค่า x,y ของเคอร์เซอร์ นำค่า x มาตรวจว่าอยู่ในส่วนใดของเมนู Cmode=0 คือส่วนตารางซ้ายมือ Cmode=1 เมื่อค่า x>40 คือส่วนของการวาดภาพและสี แต่แยกส่วนการตรวจสีออกมาเป็น Cmode=2 เมื่อค่า x>590 นำค่า Cmode ที่ได้ไปตรวจการทำงานในขั้นต่อไป

7. บรรทัดที่ 119-151 เป็นการทำงานในกรณีที่ค่า Cmode=1 ซึ่งก็คือส่วนของการวาดภาพตามเงื่อนไขที่ได้กำหนดไว้คือ Mode=1 ถึง 5 (ลงจุดไปจนถึงลากเส้นตรง) เนื่องจากการตรวจหา Cmode นั้น จะได้ค่า x ระหว่าง 41-589 ดังนั้นจึงต้องมีเงื่อนไขการลงจุดว่าจะทำต่อเมื่อค่า x<580 เท่านั้น โดยจะตรวจซ้ำว่ากำลังอยู่ในค่า Mode ใดเพื่อที่จะทำงานได้ถูกต้อง เช่น เมื่อ mode=1 ก็จะทำการลงจุด ณ ตำแหน่ง x,y ที่หาได้จากชุดคำสั่ง FindXY ตามคำสั่งในบรรทัดที่ 123 ตัวอย่างส่วนนี้ในรูปที่ 5 คือข้อความว่า การลงจุด

8. การลงจุดขนาด 3×3 จุด (mode=2) จะทำงานตามคำสั่งบรรทัดที่ 124-128 ซึ่งเราจะต้องสั่งให้หยุดการแสดงเคอร์เซอร์เสียก่อน เพื่อจะได้ลงจุดทับลง ณ ตำแหน่งของเคอร์เซอร์ได้ เมื่อลงจุดตามต้องการแล้วจึงสั่งให้แสดงเคอร์เซอร์ใหม่ (เคอร์เซอร์ของเมาส์จะทำงานด้วยซอฟต์แวร์ที่เรารันในตอนแรก โดยจะจำภาพ ณ ตำแหน่งที่เคอร์เซอร์จะแสดงไว้ก่อน เพื่อที่จะนำมาแสดงซ้ำเมื่อเคอร์เซอร์เลื่อนไปที่อื่น) ถ้าวาดภาพโดยไม่สั่งให้หยุดแสดงเคอร์เซอร์ของเมาส์เสียก่อนจะทำให้ได้ภาพที่ไม่สมบูรณ์

กรณีนี้ต่างจากการลงจุดเพราะการลงจุดนั้นจะลง ณ ปลายลูกศรของเคอร์เซอร์แสดงอยู่ (สูงกว่าปลายลูกศรเล็กน้อย) จึงไม่จำเป็นต้องสั่งให้หยุดแสดงเคอร์เซอร์ก่อนลงจุดก็ได้ ตัวอย่างภาพของส่วนนี้ในรูปที่ 5 คือข้อความว่า สวัสดี

9. บรรทัดที่ 129-137 เป็นการทำงานในกรณีสร้างวงกลม (mode=3) เริ่มด้วยการหยุดแสดงเคอร์เซอร์ของเมาส์ คำนวณหาค่าระยะในแกนนอน (x1), ระยะในแกนตั้ง (y1) โดยกำหนดค่าให้เป็นค่าบวก นำค่าทั้งสองมาคำนวณหารัศมีของวงกลมด้วยคำสั่ง Radius= round (sqrt ((sqr(x1)+sqr(y1))); เพื่อให้รัศมีของวงกลมเป็นจริงตามตำแหน่งที่เคอร์เซอร์ปรากฏ จึงต้องนำค่า xasp, yasp มาใช้ในการคำนวณหาค่า y ด้วย (ตามบรรทัดที่ 131)

ขอให้สังเกตเทคนิคการใช้คำสั่ง setviewport ที่ผู้เขียนนำมาช่วยในกรณีนี้ โดยมีจุดประสงค์ที่สำคัญ คือ ไม่ให้วงกลมที่วาดนั้นแสดงเลยออกไปนอกกรอบภาพที่กำหนด (กำหนดเป็น clipon) นั่นคือกำหนดกรอบให้วาดอยู่ในขอบเขต 41, 31, 579, 299 แต่เพื่อให้วงกลมยังแสดง ณ ตำแหน่งเดิมจึงต้องเลื่อนจุดศูนย์กลางไปเป็นจุด Cx-41, Cy-31 (เพราะตอนนี้มุมบนซ้ายเลื่อนมาที่ 41, 31) เมื่อวาดวงกลมเสร็จแล้วต้องคืนกรอบจอภาพให้เท่ากับกรอบจอภาพปกติ

คำสั่ง setviewport นี้เป็นคำสั่งหลอกให้ปาสคาลมองเห็นกรอบจอภาพผิดไปจากความจริง แต่ผู้ดูบนจอภาพยังคงเห็นจอภาพตามปกติ การเลื่อนจุดให้สัมพันธ์กันจึงจะทำให้ผู้ดูบนจอภาพมองเห็นภาพตามความจริง ขอให้สังเกตการทำงานนี้บนจอภาพจะเข้าใจเรื่อง setviewport ได้ดีขึ้น

10. บรรทัดที่ 138-143 การสร้างรูปสี่เหลี่ยมคำสั่งนี้คงทำความเข้าใจได้ไม่ยาก

11. บรรทัดที่ 144-148 การลากเส้นตรง

จบการทำงานในส่วนของการวาดภาพ (เมื่อ Cmode=1) เพียงแค่นี้

12. กรณีที่ Cmode=0,2 คือช่วงการกดปุ่มซ้ายเมื่ออยู่ในตารางซ้ายมือหรือขวามือ จะเริ่มต้นด้วยการตรวจว่าเมื่อกดปุ่มของเมาส์นั้นกำลังอยู่ที่ตารางลำดับที่เท่าใด (นับจากข้างบนลงล่าง) ซึ่งคือคำสั่งในบรรทัดที่ 153-157 โดยจะได้ตัวแปร no มาใช้งาน

การทำงานในช่วงถัดไปก็จะเป็นการตรวจค่า no ตามค่า Cmode ว่าเป็น 0 หรือ 2

13. ถ้า Cmode=0 จะใช้คำสั่งบรรทัดที่ 161-178 โดยที่ no=1 ถึง 11 เป็นการกำหนดรูปแบบของพื้นตาม บรรทัดที่ 162-164 ได้ค่าของพื้นตามตัวแปร patt ไปใช้งานโดยทำการกำหนดรูปแบบพื้นด้วยสีพื้นตามตัวแปร color นั่นคือถ้าต้องการสีพื้นเป็นสีใดก็ต้องเลื่อนเมาส์ไปเลือกสีที่ต้องการก่อนแล้วมาเลือกรูปแบบพื้นทีหลัง

แต่ถ้าค่า no>11 จะเป็นการเลือกรูปแบบของการวาดภาพ (หาจาก no-11) ทำให้ได้ตัวแปร mode ตามต้องการดังคำสั่งในบรรทัดที่ 166-178 ซึ่งจะมีการแสดงภาพของส่วนนั้น ๆ แบบกลับขาวดำพร้อมกับแสดงข้อความที่มุมบนของกรอบวาดภาพด้วย โดยมีการกำหนดค่าให้กับตัวแปร Cx,Cy ด้วย

14. ถ้า Cmode=2 (คือการเลือกสี) จะกำหนดค่าสีให้ตัวแปร color พร้อมกับสั่งกำหนดสีให้กับระบบโดยการกำหนดนี้จะเกิดขึ้นเฉพาะกรณีที่ค่า no น้อยกว่า 16 เท่านั้น

จบส่วนการทำงานในกรณีที่มีการกดปุ่มซ้ายของเมาส์เพียงแค่นี้

15. บรรทัดที่ 190-203 คือส่วนสุดท้ายซึ่งเป็นกรณีที่ regs.BX=4 คือการกดปุ่มกลางของเมาส์ ซึ่งจะทำงานเฉพาะกรณีที่ mode มีค่า 3-5 (วงกลม, สี่เหลี่ยม และเส้นตรง) จะเป็นการกำหนดจุดศูนย์กลางหรือมุมแรกของสี่เหลี่ยมหรือจุดแรกของเส้นตรง ค่านี้จะเปลี่ยนใหม่ทุกครั้งที่มีการกดปุ่ม โดยจะตรวจหา x,y และให้ค่านี้แก่ตัวแปร Cx,CxOld,Cy,CuOld และลงจุดให้เห็นหนึ่งจุด

การวาดภาพทั้งหมดจะสิ้นสุดลงเมื่อกดปุ่มด้านขวา (ค่า regs.BX=2) ต่อจากนี้ก็เป็นการเขียนส่วนของโปรแกรมหลักเพื่อนำชุดคำสั่งทั้งหมดมาทำงานได้ดังรูปที่ 4 เมื่อนำมารวมกันเป็นซอฟต์เมล์จะได้ไฟล์ชื่อ mouse.pas มีรายละเอียดดังนี้

1. บรรทัดที่ 206 การเข้าสู่กราฟิกและคำนวณหาค่าของ xasp, yasp

2. กำหนดสีและให้ตัวแปรบูลีน mouse เป็นเท็จ

```
204: {========================== Main ====================}
205: begin
206:    OpenGraph;          getaspectRatio(Xasp,Yasp);
207:    setcolor(14);       Mouse:=false;
208:    GetMem(Pic,imagesize(0,0,30,40));
209:    CheckMouse;
210:    if Mouse then
211:      begin
212:         PicByMouse;
213:         Mouseoff;
214:      end
215:       else ch:=readkey;
216:    freeMem(Pic,imagesize(0,0,30,40));
217:    CloseGraph;
218: end.
```

รูปที่ 4 ส่วนประกอบโปรแกรมหลักในการวาดภาพด้วยเมาส์

รูปที่ 5 ภาพตัวอย่างที่วาดได้จากโปรแกรม mouse.pas

3. กำหนดหน่วยความจำฮีพให้กับตัวแปร pic

4. ตรวจการติดตั้งเมาส์

5. ถ้ามีการติดตั้งเมาส์ก็จะเข้าสู่ส่วนของการวาดภาพด้วยเมาส์

6. ถ้าไม่มีการติดตั้งเมาส์ก็จะรอกดแป้น

7. ยกเลิกหน่วยความจำฮีพของตัวแปร pic

8. ออกจากกราฟิกโหมดด้วยคำสั่ง closegraph

ตัวอย่างการวาดภาพในการกำหนดรูปแบบพื้นและสี อาจเป็นดังนี้

1. เลื่อนเมาส์ไปเลือกการวาดรูปสี่เหลี่ยมกดปุ่มซ้ายมือ

2. เลื่อนเมาส์มาอยู่ในกรอบที่จะวาดภาพกดปุ่มกลางกำหนดมุมแรกของรูปสี่เหลี่ยม

3. เลื่อนเมาส์ไปยังด้านขวาเพื่อเลือกสีม่วงกดปุ่มซ้ายมือ

4. เลื่อนเมาส์ไปยังด้านซ้ายเพื่อเลือกพื้นแบบที่ 9 กดปุ่มซ้ายมือเป็นการเลือกพื้นแบบที่ 9 ที่แสดงพื้นสีม่วงและมีเส้นกรอบสี่เหลี่ยมเป็นสีม่วงด้วย

5. เลื่อนเมาส์ไปยังด้านขวาเพื่อเลือกสีเหลืองกดปุ่มซ้ายมือ มาถึงขั้นนี้จะได้พื้นสีม่วงที่มีเส้นกรอบเป็นสีเหลือง

6. เลื่อนเมาส์ไปยังส่วนการวาดภาพเลือกมุมที่สองของสี่เหลี่ยมกดปุ่มซ้ายมือจะได้รูปสี่เหลี่ยมพื้นสีม่วงกรอบสีเหลือง

7. ถ้ายังไม่มีการเลือกรูปแบบพื้นหรือสีก็จะได้รูปสี่เหลี่ยมแบบเดิมตลอดไป

8. อาจเลื่อนไปเลือกสีอีกได้ซึ่งจะเป็นการเปลี่ยนสีของกรอบ แต่ถ้าทำตามข้อ 3-4 ก็จะเลือกสีและรูปแบบพื้นใหม่

9. การสร้างรูปแบบอื่นก็ใช้แนวคิดเดียวกันนี้

คงจบส่วนของโปรแกรมทั้งหมดเพียงแค่นี้ โปรแกรมนี้เป็นเพียงตัวอย่างของการใช้เมาส์ตัวอย่างหนึ่งเท่านั้น ผู้อ่านคงจะได้แนวทางในการนำไปพัฒนาโปรแกรมของตนเอง เพื่อให้ได้โปรแกรมที่ดียิ่งขึ้นต่อไป ตัวอย่างเช่น ผู้เขียนทิ้งส่วนของการลบภาพที่ไม่ต้องการไว้เพื่อให้ผู้อ่านลองพัฒนาเพิ่มเติมเอง โดยขอให้แนวคิดไว้ว่า ถ้าสีในขณะนั้นเป็นสีหมายเลข 0 ก็จะสามารถใช้คำสั่งเดิมมาเป็นการลบภาพได้ หวังว่าคงคิดกันออกนะครับว่าจะเพิ่มที่ตรงไหน ขอบอกเพียงว่าผู้เขียนกำหนดตัวแปรไว้เพื่อรับค่าสีหมายเลข 0 ไว้แล้ว โดยตรวจจากเศษของตัวแปรนี้หารด้วยเลขคงที่ค่าหนึ่ง ก็จะได้สีหมายเลข 0-15 ตามต้องการ (ต้องวาดภาพตารางของสีนี้เพิ่มเติมด้วย)

รูปที่ 5 เป็นตัวอย่างหนึ่งที่ได้จากการวาดภาพที่ได้จากโปรแกรมนี้

สิ่งที่ผู้อ่านต้องเขียนโปรแกรมเพิ่มเติม คือการเก็บภาพที่วาดได้แล้วนั้นลงไว้ในแผ่นบันทึก ซึ่งคงจะทำไม่ยากนัก โอกาสหน้าผู้เขียนอาจจะนำเสนอเทคนิคของการเก็บภาพลงแผ่นบันทึกมาเป็นตัวอย่างต่อไป